Practical – 2

Aim: Implement and analyse algorithms given below:

2.1) Binary Search

2.2) Insertion Sort

Program Code:

```
import java.util.*;
import java.time.*;
public class trial
{
   public static int BinarySearch(int a[],int key)
   {
      int start=0;
      int end=a.length-1;
      while(start<=end)
      {
         int mid=start+(end-start)/2;
         if(a[mid]==key)
         {
            return mid;
         }
         else if(key>a[mid])
         {
            start=mid+1;
         }
         else
```

```
        {
            end=mid-1;
        }
    }
    return -1;
}
public static void InsertionSort(int a[])
{
    for(int i=1 ; i<a.length ; i++)
    {
        int temp=a[i];
        int j=i-1;
        while(j>=0 && temp<a[j])
        {
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=temp;
    }
}
public static void main(String[] args)
{
    int key=30;
    int a[]={10,20,30,40,50,60,70,80,90,100};
    System.out.println();
```

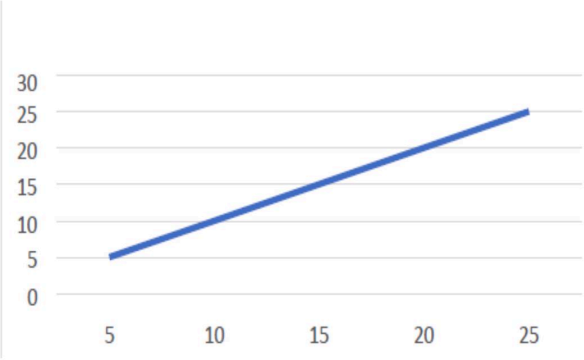
```
        long begin_time=System.currentTimeMillis();
        System.out.println(key+" is found at "+BinarySearch(a,key)+" using Binary Search\n");
        long end_time=System.currentTimeMillis();
        System.out.println("time taken by Binary Search= "+(end_time-begin_time));


        System.out.println("\nQuick Sort");
        int b[]={5,2,6,7,1,3,4};
        Instant inst11 = Instant.now();
        InsertionSort(b);
        Instant inst22 = Instant.now();
        System.out.println("Elapsed Time: "+ Duration.between(inst11, inst22).toString());
        display(b);
}
```

OUTPUT:

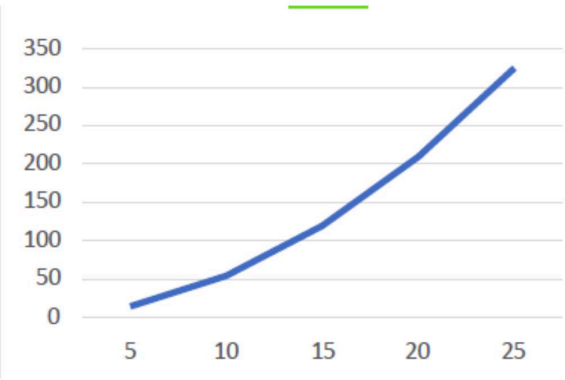
```
PROBLEMS 92  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  COMMENTS

PS D:\Probin's Work\DSA> javac mytree.java
PS D:\Probin's Work\DSA> java mytree

30 is found at 2 using Binary Search

time taken by Binary Search= 6 ms
```

```
Insertion Sort
Elapsed Time: PT0.000004795S
Array:
 1 2 3 4 5 6 7
```

Insertion Best Case



Insertion Average Case



CONCLUSION: From this practical I learned about binary searching which is faster than linear search which is suitable for large datasets. Time Complexity is O(log$n$). and insertion sort technique which is adaptive in nature which is suitable for datasets that is already partially sorted. Time Complexity is O(n^2).

Staff Signature:

Grade:

Remarks by the Staff: